(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 017 178 A1

(12) EUROPEAN PATENT APPLICATION

published in accordance with Art. 158(3) EPC

(43) Date of publication:
05.07.2000 Bulletin 2000/27

(21) Application number: **99919618.1**

(22) Date of filing: **17.05.1999**

(51) Int. Cl.[7]: **H03M 13/23**

(86) International application number:
PCT/JP99/02553

(87) International publication number:
WO 99/62183 (02.12.1999 Gazette 1999/48)

(84) Designated Contracting States:
DE ES FR GB IT

(30) Priority: **28.05.1998 JP 14676298**
25.08.1998 JP 23898798

(71) Applicant: **Sony Corporation**
Tokyo 141-0001 (JP)

(72) Inventors:
- **MIYAUCHI, Toshiyuki**
  Sony Corporation
  Tokyo 141-0001 (JP)
- **HATTORI, Masayuki**
  Sony Corporation
  Tokyo 141-0001 (JP)

(74) Representative:
Melzer, Wolfgang, Dipl.-Ing. et al
Patentanwälte
Mitscherlich & Partner,
Sonnenstrasse 33
80331 München (DE)

(54) SOFT OUTPUT DECODER FOR CONVOLUTION CODE AND SOFT OUTPUT DECODING METHOD

(57) A soft output decoding method and apparatus for convolutional codes. After computing the number of states times Iβ ($\beta_t \sim \beta_{t-D+1}$) for a truncated length, the soft output outside the truncated length is sequentially computed, as next following Iβ ($\beta_{t-D} \sim \beta_{t-2D+1}$) outside the next following truncated length is computed, at the same time as Iβ of the next truncated length is sequentially computed. In this manner, a decoder 4 performs computation of Iβ within the truncated length and computation of Iβ retrogressive by not less than the truncated length, as parallel processing, as a result of which the computation of Iβ per clock is the number of states ×2 to reduce the volume of computation to expedite the decoding.

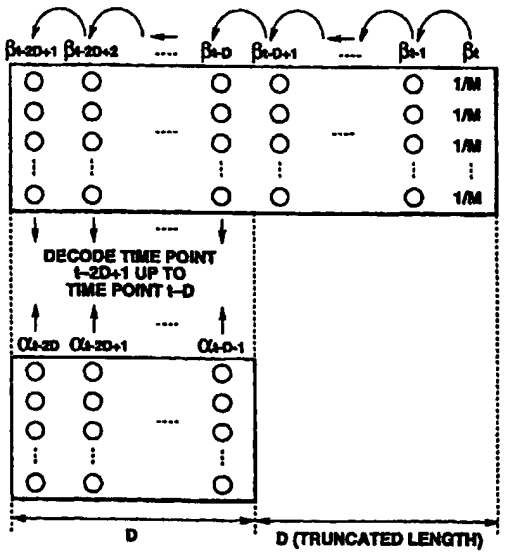

FIG.7

EP 1 017 178 A1

Description

Technical Field

[0001] This invention relates to a soft output decoding method and apparatus for convolutional codes, applied with advantage to, for example, a satellite broadcast reception device. More particularly, it relates to a soft output decoding device in which the probability information is stored on a recording medium a length not less than a truncated length and in which updating of the probability information within the truncation length and computations of the soft output outside the truncated length are executed in parallel to enable high-speed operations.

Background Art

[0002] As a decoding method for minimizing the symbol error ratio following decoding of the convolutional codes, there is known a BCJR algorithm from Bahl, Cocke, Jelinek and Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", IEEE Trans. Information. Theory, Vol.1T-20, pp. 284 to 287, Mar. 1974. In the BCJR algorithm, it is not each symbol, but the likelihood of each symbol, that is outputted as the decoding result. This output is termed soft output.

[0003] In these years, researches are being conducted towards reducing the symbol error rate by using soft outputs as decoded output of the inner code of the conjugated code or as outputs of each reiteration of the iterative decoding method. As a decoding method, suited to this purpose, a BCJR algorithm is stirring up notice.

[0004] The contents of the BCJR algorithm are hereinafter explained in detail.

[0005] The BCJR algorithm outputs the likelihood of each symbol, instead of outputting each symbol, as a decoding result. The BCJR algorithm is used when convolutional encoding the digital information is convolutional-encoded by a convolutional encoder and the resulting data string obtained on convolutional encoding is observed over a non-storage communication route.

[0006] M states (transition states) representing the contents of shift registers of a convolutional encoder are expressed as m(0, 1, • • •, M|1). The state at time t is S_t, an input at t is I_t, an output at time t is X_t, and an output sequence is $X_t^{t'} = X_t, X_{t+1}, \cdots, X_{t'}$. The transition probability between respective states $p_t(m|m')$ is represented by the following equation (1):

$$p_t(m|m') = Pr\{S_t = m|S_{t-1} = m'\} \quad (1).$$

[0007] It is noted tat Pr{A|B} is a conditional probability of occurrence of A, under a condition that an event B has occurred, while Pr{A;B} is a probability of occurrence of both A and B. It is also assumed that the convolutional codes by the convolutional encoder starts with the state $S_0 = 0$ and outputs X_1^{τ} to terminate at $S_\tau = 0$.

[0008] The noisy non-storage communication route receives an output X_1^{τ} as an input and outputs Y_1^{τ}. It is assumed that an output sequence $Yt^{t'} = Y_t, Y_{t+1}, \cdots, Y_{t'}$. Meanwhile, the transition probability of the non-storage communication route can be defined, for all t ($1 \le t \le \tau$), by a function $R(\cdot|\cdot)$ satisfying the following equation (2):

$$Pr\{Y_1^{t}|X_1^{t}\} = \prod_{j=1}^{t} R(Y_j|X_j) \quad (2).$$

[0009] Therefore, if the probability λ_t is defined as in the following equation (3):

$$\Lambda t = \begin{matrix}\{Pri_t=1|Y_1^{\tau}\} \\ \{Pri_t=0|Y_1^{\tau}\}\end{matrix} \quad (3)$$

this probability λ_t represents the likelihood of the input information at time t when Y_1^{τ} is received, such that this probability λ_t is the soft output to be found.

[0010] The BCJR algorithm defines the probabilities α_t, β_t and γ_t as indicated by the following equations (4) to (6):

$$\alpha_t(m) = Pr\{S_t = m; Y_1^{t}\} \quad (4)$$

$$\beta_t(m) = Pr\{T_{t+1}^{\tau}|S_t = m\} \quad (5)$$

$$\gamma_t(m', m, i) = Pr\{St = m; Y_t; i_t = i|S_{t-1} = m'\} \quad (6).$$

[0011] The contents of α_t, β_t and γ_t are explained with reference to Fig. 1 which illustrates the relation between the respective probabilities. It is noted that α_{t-1} corresponds to the probability of passing through respective states at time t-1, as computed based on a reception word as from the encoding starting state S0 = 0, while β_t corresponds to the probability of passing through respective states at time T as computed in the reverse sequence to the chronological sequence based on the reception word as from the encoding end state $S\tau = 0$, and γ_t corresponds to the reception probability of respective branches in transition through respective states at time t as computed based on the reception word and the input probability at time t.

[0012] With the aid of α_t, β_t and γ_t, the soft output λ_t can be represented by the following equation (7):

$$\gamma_t = \frac{\sum_{m=0}^{M-1}\sum_{m'=0}^{M-1}\alpha_{t-1}(m')\chi_t(m',m,1)\beta_t(m)}{\sum_{m=0}^{M-1}\sum_{m'=0}^{M-1}\alpha_{t-1}(m')\chi_t(m',m,0)\beta_t(m)} \quad \cdots(7).$$

[0013] It is noted that, for t = 1, 2, • • •, τ, the following equation (8) holds:

$$\alpha_t(m) = \sum_{m=0}^{M-1}\sum_{t'=0}^{1}\alpha_{t-1}(m')\chi_t(m',m,i) \quad (8).$$

where $\alpha_0(0) = 1$, $\alpha_0(m) = 0$ $(M \neq 0)$.

[0014] Similarly, the following equation (9):

$$\beta_t(m) = \sum_{m=0}^{M-1}\sum_{i=0}^{1}\beta_{t+1}(m')\chi_{t+1}(m',m,i) \quad (9)$$

holds for t = 1, 2, • • •, τ - 1, where $\beta_\tau(0) = 1$, $\beta_\tau(m) = 0$ $(m \neq 0)$.

[0015] For γ_t, the following equation (10) holds:

$\gamma_t(m',m,i) = P_t(m|m')R(Y_t, X)$ in case of transition from m' to m for an input i (X being its output); and (10).

$\gamma_t(m',m,1) = 0$ in case of non-transition from m' to m for the input i

[0016] Based on the above equation, the BCJR algorithm finds the soft output λ_t in accordance with the following sequence (a) to (c):

(a) each time Y_t is received, $\alpha_t(m)$, $\gamma_t(m',m,i)$ is computed using the equations (8) and (10);
(b) after reception of the entire sequence Y_1^τ, $\beta_t(m)$ is computed for respective states m for the totality of time points t, using the equation (9); and
(c) α_t, β_t and γ_t, computed at (a) and (b), are substituted into the equation (7) to compute the soft output λ_t at each time point t.

[0017] Meanwhile, the above-described BCJR algorithm suffers from the problem that the volume of computation

is considerable because of the product computations involved and that continuous data cannot be received because of the necessity of terminating the codes.

[0018] In order to combat these problems, a Max-Log-BCJR algorithm and a log-BCJR algorithm have been proposed as techniques for diminishing the computation volume in Robertson, Villebrun and Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Domain", in IEEE Int. Cof. On Communications, pp. 1009 to 1013, June 1995,while the SW-BCJR algorithm for performing sliding window processing has been proposed as a technique for receiving continuous data in Benedetto and Montorsi, "Soft-Output Decoding Algorithm in Iterative Decoding of Turbo Codes", TDA Progress Report 42-124, Feb. 1996.

[0019] The contents of these algorithms are hereinafter explained.

[0020] First, the contents of the Max-Log-BCJR algorithm and the log-BCJR algorithm are explained.

[0021] The Max-Log-BCJR algorithm represents the probabilities α_t, β_t and γ_t, λ_t by a modulo 2 logarithm (natural logarithm) and substitutes the logarithmic sum computations for the product processing of the probabilities, as shown in the following equation (11), while approximating the logarithmic sum computations by the logarithmic maximum value computations, as shown in the following equation (12). Meanwhile, max(x, y) is a function which selects a larger one of x or y.

$$\mathrm{Log}(e^x \cdot e^y) = x + y \tag{11}$$

$$\mathrm{Log}(e^x + e^y) = \max(x, y) \tag{12}.$$

[0022] For simplifying the explanation, logarithms of α_t, β_t and γ_t, λ_t are set as $I\alpha_t$, $I\beta_t$, $I\gamma_t$, $I\lambda_t$, respectively, as shown by the following equations (13) to (15):

$$I\alpha_t(m) = \log(\alpha_t(m)) \tag{13}$$

$$I\beta_t(m) = \log(\beta_t(m)) \tag{14}$$

$$I\gamma_t(m) = \log(\gamma_t(m)) \tag{15}$$

where "I" denotes the modulo e logarithm.

[0023] In the Max-Log-BCJR algorithm, these $I\alpha_t$, $I\beta_t$, $I\gamma_t$ are approximated as shown by the following equations (16) to (18):

$$I\alpha_t(m) = \max_{i=0,1} \max_{m'}(I\alpha_{t-1}(m') + I\chi_t(m', m, i)) \quad \cdots(16)$$

$$I\beta_t(m) = \max_{i=0,1} \max_{m'}(I\beta_{t+1}(m') + I\chi_t(m', m, i)) \quad \cdots(17)$$

$$I\gamma_t(m', m, i) = \log(P_t(m|m')) + \log(R(Y_t, X)) \tag{18}$$

where X is an encoder output on transition from m' to m. It is noted that max m' in $I\alpha_t(m)$ and $I\beta_t(m)$ is to be found in the state m' in which transition to a state m occurs for the input i.

[0024] Similarly, $I\lambda_t$ is approximated as shown in the following equation (19), in which max m' in the first term of the right side is to be found in the state m' where transition to the state m exists for the input = 1 and in which max m' in the second term is to be found in the state m' where transition to the state m exists for the input = 0.

$$I\lambda_t = \max_{i=0,1} \max_{m'}(I\alpha_{t-1}(m') + I\chi_t(m',m,1) + I\beta_t(m)) - \max_{m=0,\cdots,m=1} \max_{m'}(I\alpha_{t-1}(m') + I\chi_t(m',m,0) + I\beta_t(m)) \quad \cdots(19).$$

[0025] Based on the above relation, the Max-Log-BCJR algorithm finds the soft output λ_t in accordance with the following sequence (a) to (c):

(a) each time Y_t is received, $I\alpha_t(m)$ and $I\gamma_t(m',m,1)$ are computed, using the equations (16) and (18);
(b) after receiving the sequence Y_1^T in its entirety, $I\beta_t(m)$ is computed, for all states m of all time points t, using the equation (17);
(c) $I\alpha_t$, $I\beta_t$ and $I\gamma_t$, computed in (a) to (c), are substituted into the equation (19) to compute the soft output $I\lambda_t$ at each time point.

[0026] Since there are included no product computations in the Max-Log-BCJR algorithm, the processing volume can be reduced significantly in comparison with the BCJR algorithm.

[0027] Meanwhile, if the probability sum computation is modified as shown in the following equation (20):

$$\mathrm{Log}(e^x + e^y) = \max(x, y) + \log(1 + e^{-|x-y|}) \quad (20)$$

the second term of the right side becomes the linear function for the variable $|x - y|$, so that, by corresponding tabulation, the logarithmic values of the sum processing can be found correctly.

[0028] The Log-BCJR algorithm substitutes the equation (20) for the equation (12) in the Max-Log-BCJR algorithm in its entirety to realize correct probability computations. As compared to the Max-Log-BCJR algorithm, the Log-BCJR algorithm is increased in processing volume. However, there is included no product computation in the Log-BCJR algorithm, such that its output is no other than the logarithmic value of the soft output of the BCJR algorithm except the quantization error.

[0029] Since the second term of the right side in the equation (20) is the linear function by the variable $|x - y|$, it is possible to obtain simple and highly accurate results of computation by, for example, tabulation. Thus, a soft output higher in accuracy can be obtained with the Log-BCJR algorithm than is possible with the Max-Log-BCJR algorithm.

[0030] The contents of the SW-BCJR algorithm are hereinafter explained.

[0031] In the BCJR algorithm, the code needs to be terminated for γ_t computation, such that continuous data cannot be received. The BCJR algorithm accords 1/M for the totality of states as an initial value of β_t, introduces the truncated length as in the case of Viterbi decoding and finds the soft output by retrograding down the time axis by a truncated length D as set (see Fig.2).

[0032] The SW-BCJR algorithm initializes α_0, as in the case of the usual BCJR algorithm, and performs the operations (a) to (e) for each time instant to find the soft output for each time instant.

(a) γ_t is found based on the received value at time t and the transition probability;

(b) $\beta_t(m)$ is initialized for the totality of states m so that $\beta_t(m) = 1/M$;

(c) $\beta_{t-1}, \cdots, \beta_{t-D}$ are computed based on $\gamma_{t-D-1}, \cdots, \gamma_t$;

(d) from β_t-D and α_{t-D-1}, as found, the soft output γ_{t-D} at time t-D is found by the following equation (21);

$$\gamma_{t-D} \quad \frac{\Sigma\alpha_{t-D-1}(m',m,1) + I\beta_{t-D}(m)}{\Sigma\alpha_{t-D-1}(m',m,0) + I\beta_{t-D}(m)} \quad (21)$$

and

(e) from α_{t-D-1} and γ_{t-D}, α_{t-D} is computed.

[0033] In the above-given thesis by Benedetto et al., there are also proposed a SW-Log-BCJR algorithm, combined from the SW-BCJR algorithm and the Log-BCJR algorithm, and a SW-Max-Log-BCJR algorithm, combined from the SW-BCJR algorithm and the Max-Log-BCJR algorithm. It is noted that the SW-Max-Log-BCJR algorithm is referred to in the thesis as SWAL-BCJR algorithm.

[0034] With use of the SW-Max-Log-BCJR algorithm or the SW-Log-BCJR algorithm, it is possible to receive continuous data to find a soft output. However, in these algorithms, in contradistinction from the case of decoding terminated codes, it is necessary to find the number of states multiplied by Y_t for the truncated length in order to find one decoded output, with the result that a large processing volume is involved for mounting even though the product computations are nit involved.

[0035] The SW-Max-Log-BCJR algorithm or the SW-Log-BCJR algorithm has a drawback that, while it is possible to receive the convolutional encoded and transmitted continuous data to find the soft output, the processing volume for producing a code output is increased to render high-speed processing difficult.

[0036] On the other hand, with the SW-Log-BCJR algorithm, combined from the SW-BCJR algorithm and the Log-BCJR algorithm, or the SW-Max-Log-BCJR algorithm, combined from the SW-BCJR algorithm and the Max-Log-BCJR algorithm, also termed the SWAL-MAP algorithm, it is possible to diminish the processing volume to find the soft output of continuous data.

[0037] However, with these algorithms, it is necessary to retrograde down the time axis by the truncated length D to produce a soft output to find β for the number of states times the truncated length D, with the result that a tremendous processing volume is required despite the fact that product processing is not involved in the processing.

Disclosure of the Invention

[0038] It is therefore an object of the present invention to provide a method and apparatus for decoding a soft output of convolutional codes with a high-speed operation.

[0039] It is another object of the present invention to provide a method and apparatus for decoding a soft output of convolutional codes by a simplified structure.

[0040] According to the present invention, when finding the probability information at each transition state of the convolutional code, and computing and outputting a soft output using the probability information, the probability information is partitioned in terms of a pre-set truncated length as a unit and stored. The updating of the probability information in the truncated length and the computation of the soft output outside the truncated length are executed in parallel.

[0041] According to the present invention, a smaller processing volume per clock or a smaller amount of accessing to the memory suffices to enable a high-speed processing.

[0042] Thus, in one aspect of the present invention, there is provided a soft output decoding apparatus for convolutional codes probability computing means for finding the probability information in each transition state of the convolutional codes, probability storage means for storing the probability information as found by the probability computing means in a recording medium, and soft output computing means for finding a soft output using the probability information stored in the recording medium. The probability storage means stores the probability information in an amount not less than a truncated length on the recording medium, and the updating of the probability information within the truncated length by the probability storage means and the computation of the soft output outside the truncated length by the soft output computing means are carried out in parallel.

[0043] In another aspect of the present invention, there is provided a soft output decoding method for convolutional codes including a first step of finding the probability information at each transition state of the convolutional codes, a second step of storing the probability information as found in the first step in the recording medium in the first step a length not less than a truncated length, and a third step of finding a soft output using the probability information stored in the recording medium in the second step. The updating of the probability information within the truncated length in the second step and the computation of the soft output outside the truncated length in the third step are carried out in parallel.

Brief Description of the Drawings

[0044]

Fig.1 illustrates the contents of α_t, β_t and γ_t in the BCJR algorithm.

Fig.2 illustrates the contents of the SW-BCJR algorithm.

Fig.3 is a block diagram showing a communication model embodying the present invention.

Fig.4 is a block diagram showing the structure of a convolutional encoder in the communication model.

Fig.5 shows the trellis of the convolutional encoder.

Fig.6 is a block diagram showing the structure of a decoder in the communication model.

Fig.7 illustrates the sequence of soft output computation in the communication model.

Fig.8 is a block diagram showing the structure of an $I\gamma$ computation storage circuit in the decoder shown in Fig.6.

Fig.9 is a timing chart for illustrating the operation of a RAM constituting the $I\gamma$ computation storage circuit.

Fig.10 is a block diagram showing the structure of the $I\alpha$ computation storage circuit in the decoder shown in Fig.6.

Fig.11 is a block diagram showing the structure of an $I\alpha$ computation circuit in the $I\alpha$ computation storage circuit.

Fig.12 is a block diagram showing the structure of an addition comparison selection circuit within the $I\alpha$ computation circuit.

Fig.13 is a timing chart for illustrating the operation of a register and a RAM making up the $I\alpha$ computation storage circuit.

Fig.14 is a bd showing the structure of the $I\beta$ computation storage circuit.

Fig.15 is a block diagram showing the structure of an $I\beta$ computing circuit in the $I\beta$ computation storage circuit.

Fig.16 is a block diagram for illustrating the structure of an addition comparison selection circuit in the $I\beta$ computation circuit.

Fig.17 is a timing chart for illustrating the operation of the register etc making up the $I\beta$ computation storage circuit.

Fig.18 is a block diagram showing the structure of a soft output computation circuit in the decoder.

Fig.19 is a block diagram showing the structure of an $I\lambda_1$ computation circuit in the soft output computation circuit.

Fig.20 is a block diagram showing die structure of an $I\lambda_0$ computation circuit in the soft output computation circuit.

Fig.21 is a timing chart for illustrating the operation of the soft output computation circuit.

Figs.22A to D illustrate the contents of a memory management in the $I\gamma$ computation circuit.

Fig.23 is a timing chart of the memory management.

Fig.24 is a block diagram showing the structure of an addition comparison selection circuit associated with the SW-Log-BCJR algorithm.

Fig.25 is a block diagram showing an alternative structure of the $I\gamma$ computation storage circuit in the decoder.

Fig.26 is a timing chart for illustrating the operation of the $I\gamma$ computation storage circuit.

Fig.27 is a block diagram showing an alternative structure of an $I\beta$ computation storage circuit in the decoder.

Fig.28 is a timing chart for illustrating the operation of the $I\beta$ computation storage circuit.

Best Mode For Carrying Out the Invention

[0045] Referring to the drawings, preferred embodiments of the present invention will be explained in detail.

[0046] The present invention is applied to a communication model 1 configured as shown for example in Fig.3. This communication model 1 convolutionally encodes the digital information D0 by a convolutional encoder 2 to send the convolutionally encoded data string to a decoder 4 over a noisy non-storage communication route 3 to decode the soft output of the convolutionally encoded data string.

[0047] The convolutional encoder 2 in this communication model 1 is a 1:2 encoder for outputting 2 bits for a 1-bit input. Referring to Fig.4, the convolutional encoder 2 is made up of an input terminal 21, fed with a 1-bit input i_t, output terminals 22a, 22b outputting a 2-bit output X_t, three exclusive-OR circuits (EX • OR circuits) 23 to 25 and two registers 26, 27.

[0048] The input terminal 21 is connected to the output terminal 22a and to an input side of the EX • OR circuit 23, an output of which is connected to the input side of the resistor 26 and to the input side of the EX • OR circuit 24. An output side of the EX • OR circuit 24 is connected to the output terminal 22b, while an output side of the resistor 26 is connected to an input side of the resistor 27 and to an input side of the EX • OR circuit 25. An output of the resistor 27 is connected to the input of the EX • OR circuit 24 and to an input of the EX • OR circuit 25, an output side of which is connected to the input side of the EX • OR circuit 23.

[0049] In this convolutional encoder 2, the 1-bit input i_t, sent to the input terminal 21, is directly outputted at the output terminal 22a, while being inputted to the EX • OR circuit 23. The EX • OR circuit 23 sends an exclusive-OR output between the input i_t and the output of the EX • OR circuit 24 via register 26 and registers 26, 27 to the EX • OR circuit 25. An exclusive-OR output of the EX • OR circuit 24 is fed back to the EX • OR circuit 23. This EX • OR circuit 23 also routes the exclusive-OR output between the input i_t and the output of the EX • OR circuit 25 directly and also via the resistors 26, 27 to the EX • OR circuit 25. The EX • OR circuit 25 outputs an exclusive-OR output between the exclusive-OR output of the EX • OR circuit 23 and an output of the resistor 27 as another bit at the output terminal 22b.

[0050] In the above-described configuration of the convolutional encoder 2, if the 1-bit input i_t is routed to the input terminal 21, a 2-bit output sequence X_t is outputted at the output terminals 22a, 22b. Fig.5 shows the trellis of the convolutional encoder 2, with the number of states being 4.

[0051] The decoder 4 in this communication model 1 is a decoder which is based on the SW-Max-Log-BCJR algorithm and which is associated with the convolutional encoder 2 having a constraint length of 3 shown in Fig.4.

[0052] This decoder 4 processes a received value Y_t of the encoder 2, received from the non-storage communication route 3, with a truncated length D = 4, to output a soft output λ_t. Referring to Fig.6, the decoder 4 includes a controller 41 for controlling the entire operation, input terminals 42y, 43p1 and 42p2, fed with the received value Y_t, a priori probability value $Pr_1 = \log Pr\{i_t = 0\}$ and $Pr_2 = \log Pr\{i_t = 1\}$, respectively, an Iγ computing circuit 43, an Iα computation storage circuit 44, an Iβ computation storage circuit 45, a soft output computation circuit 46 and an output terminal 47 for outputting a soft output $I\lambda_t$.

[0053] In contradistinction from the usual SW-Max-Log-BCJR algorithm, this decoder 4 does not compute $I\beta_t$ for the number of states × truncated length for decoding for one time juncture. That is, the decoder 4 computes Iβ (shown by $\beta_t \sim \beta_{t-D+1}$), then sequentially computes soft output outside the truncated length, as it computes Iβ outside the truncated length (shown by $\beta_{t-D} \sim \beta_{t-2D+1}$), and sequentially computes Iβ for the next truncated length. Thus, the decoder 4 performs, in parallel, the computation of Iβ in the truncated length and Iβ retrogressive down the time axis by not less than the truncated length, with the computation of Iβ per clock being the number of states × 2.

[0054] The Iγ computation storage circuit 43 is fed from the controller 41 with a control signal Scγ and with the received value Y_t, a priori probability values Pr_1, Pr_2, from the input terminals 42y, 43p1 and 42p2, respectively. The Iγ computation storage circuit 43 uses the received value Y_t, and a priori probability values Pr_1, Pr_2 to compute and store Iγ in accordance with the equation (18) every received value Y_t. The Iγ computation storage circuit 43 then routes Iγ to the computation storage circuit 44, Iβ computation storage circuit 45 and to the soft output computation circuit 46 in a sequence proper for respective processing operations.

[0055] The Iγ computation storage circuit 43 operates as first computation means for computing the first probability γ as determined by the code pattern and by the received value. It is noted that Iγ sent from the Iγ computation storage circuit 43 to the Iα computation storage circuit 44 is depicted as Iγ(α), Iγ sent from the Iγ computation storage circuit 43 to the Iβ computation storage circuit 45 is depicted as $I\gamma(\beta_1)$, $I\gamma(\beta_2)$ whilst Iγ sent from the Iγ computation storage circuit 43 to the soft output computation circuit 46 is depicted as Iγ(λ).

[0056] The Iα computation storage circuit 44 is fed from the controller 41 with a control signal Scα, while being fed from the Iγ computation storage circuit 43 with Iγ(α). This Iα computation storage circuit 44 computes and stores Iα, in accordance with the equation (16), using Iγ(α), to route this Iγ(α) to the soft output computation circuit 46 in a sequence proper to the processing. The Iα computation storage circuit 44 operates as second computing means for computing the second probability Iα from the encoding starting state chronologically to each state from one received value Y_t to another. Meanwhile, Iα sent from the Iα computation storage circuit 44 to the soft output computation circuit 46 is

depicted as Iγ(λ), while Iγ sent to the Iα computation storage circuit 44 is depicted as Iγ(α).

[0057] The Iβ computation storage circuit 45 is fed from the controller 41 with the control signal Scβ, while being fed from the Iγ computation storage circuit 43 with $I\gamma(\beta_1)$ and $I\gamma(\beta_2)$. There is a time shift of the truncation length × 2 between $I\gamma(\beta_1)$ and $I\gamma(\beta_2)$. The Iβ computation storage circuit 45 uses $I\gamma(\beta_1)$ and $I\gamma(\beta_2)$ to compute and store two sequences of Iβ in parallel, in accordance with the equation (17), to send one of the sequences of Iβ to the soft output computation circuit 46 in an order proper to the processing. This Iβ computation storage circuit 45 constitutes third computing means for computing the third probability Iβ from the truncated state to each state in a reverse sequence to the chronological sequence from one received value Yt to another. Meanwhile, Iβ sent from the Iβ computation storage circuit 45 to the soft output computation circuit 46 is termed Iβ(γ).

[0058] The soft output computation circuit 46 is fed from the Iγ computation storage circuit 43 with Iγ(λ), while being fed from the Iα computation storage circuit 44 and the Iβ computation storage circuit 45 with Iα(λ) and Iβ(λ), respectively. The soft output computation circuit 46 uses Iγ(λ), Iα(λ) and Iβ(λ) to compute Iλt in accordance with the equation (19) to chronologically re-array and output the Iγ(λ), Iα(λ) and Iβ(λ).

[0059] The specified structures of the Iγ computation storage circuit 44, Iα computation storage circuit 44, Iβ computation storage circuit 45 and the soft output computation circuit 46 are now explained.

[0060] Fig.8 shows the structure of the Iγ computation storage circuit 43. This Iγ computation storage circuit 43 includes input terminals 301Y, $301P_1$, $301P_2$ and 301S, fed with the received value Y_t, a priori probability values Pr_1, Pr_2 and with the control signal Scγ, respectively, and read-only memories (ROMs) 302a to 302d constituting a table for outputting the probability $IR(Y_t|00)$, $IR(Y_t|01)$, $IR(Y_t|10)$, $IR(Y_t|11)$ of the received value Y_t for respective states m, with the received value Y_t to the input terminal 301Y as a readout address signal.

[0061] The Iγ computation storage circuit 43 includes adders 303a, 303b for summing the a priori probability information Pr1 supplied to the input terminal $301P_1$ for the probability $IR(Y_t|00)$, $IR(Y_t|01)$ outputted by the ROMs 302a, 302b to obtain the probability Iγ[00], Iγ[01] of respective branches associated with outputs [00], [01] on the trellis, and adders 303c, 303d for summing the a priori probability information Pr1 supplied to the input terminal $301P_2$ for the probability $IR(Y_t|10)$, $IR(Y_t|11)$ outputted by the ROMs 302c, 302d to obtain the probability Iγ[10], Iγ[11] of respective branches associated with outputs Iγ[10], [11] on the trellis.

[0062] The total number of bits of the respective probabilities Iγ[00] to Iγ[11] outputted by the adders 303a to 303d is the number of bits × 2n in the case of the systematic convolutional code with the code rate of k/n. Thus, the Iγ computation storage circuit 43 sets the respective probabilities Iγ[00] to Iγ[11] with 4 bits and outputs the probabilities Iγ[00] to Iγ[11] wit a sum total of 16 bits.

[0063] The Iγ computation storage circuit 43 also includes random access memories (RAMs) 304a to 304d, adapted for sequentially storing the respective probabilities Iγ[00] to Iγ[11], outputted by the adders 303a to 303d, in accordance with the control signal Scγ and for outputting the respective probabilities in a pre-set sequence, a selection circuit 308 for selectively retrieving Iγ outputted by these RAMs 304a to 304d in accordance with the control signal γ for conversion to $I\gamma(\alpha)$ $I\gamma(\beta_1)$, $I\gamma(\beta_2)$ and Iγ(λ) and output terminals 309a to 309d for outputting these $I\gamma(\alpha)$ $I\gamma(\beta_1)$, $I\gamma(\beta_2)$ and Iγ(λ).

[0064] In the Iγ computation storage circuit 43, the probabilities $IR(Y_t|00)$, $IR(Y_t|01)$ $IR(Y_t|10)$, $IR(Y_t|11)$ of received values Y_t for respective states n are outputted from the ROMs 302a to 302d from one received value Y_t to another, such that Iγ[00], Iγ[01], Iγ[10], Iγ[11] of respective branches associated with the outputs [00], [01], [10], [11] on the trellis are obtained from the adders 303a to 303d. These probabilities Iγ[00] to Iγ[11] are sequentially stored in the RAMs 304a to 304d and read out in a pre-set sequence so as to be retrieved by the selection circuit 308 as $I\gamma(\alpha)$ $I\gamma(\beta_1)$, $I\gamma(\beta_2)$ and Iγ(γ).

[0065] Fig.9 is a timing chart showing the management of RAMs 304a to 304d. The four RAMs 304a to 304d are run in a bank configuration, having a recording capacity of 16 bits × 4 words, for storing the output data Iγ[00] to Iγ[11] of the adders 303a to 303d, in an amount corresponding to the truncated length D. The RAMs 304a to 304d thus store the probabilities Iγ[00] to Iγ[11] sequentially cyclically (Fig.9A). In Fig.9, the probabilities Iγ[00] to Iγ[11] at time points t = 1,2,3, • • • are depicted by γ1, γ2, γ3, • • •.

[0066] From the RAMs 304a to 304d, the probabilities Iγ[00] to Iγ[11] are read out with a delay corresponding to the time equal to twice the truncated length D, or 2D. From the selection circuit 308, these probabilities Iγ[00] to Iγ[11] are retrieved as the probability $I\gamma(\alpha)$ to be sent to the Iα computation storage circuit 44.

[0067] The operation of reading out the probabilities Iγ[00] to Iγ[11] for 2D, directly since the end of writing of the probabilities Iγ[00] to Iγ[11] in the RAMs 304a and 304b, in a reverse sequence to the writing sequence, and the operation of reading out the probabilities Iγ[00] to Iγ[11] for 2D, directly since the end of writing of the probabilities Iγ[00] to Iγ[11] in the RAMs 304c and 304d, in a reverse sequence to the writing sequence, are effected alternately. From the selection circuit 308, the probabilities Iγ[00] to Iγ[11], thus read out, are retrieved as the probability $I\gamma(\beta_1)$ to be supplied to the Iβ computation storage circuit 45 (Fig.9C).

[0068] On the other hand, the operation of reading out the probabilities Iγ[00] to Iγ[11] for 2D, directly since the end of writing of the probabilities Iγ[00] to Iγ[11] in the RAMs 304b and 304c, in a reverse sequence to the writing sequence, and the operation of reading out the probabilities Iγ[00] to Iγ[11] for 2D, directly since the end of writing of the probabil-